

FLEXIBLE PROCESSING HARDWARE ARCHITECTURE

RELATED APPLICATIONS

This is a Continuation-in-part of U.S. Patent Application
Serial Number 08/953,772, filed 10/17/97.

FIELD OF THE INVENTION

This invention relates to the field of flexible and
modifiable processing hardware architecture and more
particularly, to a processing hardware architecture that allows
for multiple system configurations.

BACKGROUND OF THE INVENTION

The advent of the personal computer (PC) has enabled a
variety of low cost processing systems such as image processing &
machine vision system designs based on standardized hardware
which take advantage of the technological innovation of the PC
market.

Four typical configurations of such PC-based image
processing and/or machine vision systems that are widely used
today are highlighted below.

In the simplest configuration of such as system, a frame
grabber or camera interface board or system resides on the PC's
peripheral bus and is used to acquire images from one or more
cameras into the PC's memory where the images can be accessed and

1 processed by the PC CPU. This will be referred to as the frame
2 grabber or host processing configuration since the host CPU is
3 solely responsible for all image processing operations.

4 Another class of such systems uses a plug-in processor
5 accelerator board to assist the PC CPU in performing some of the
6 computationally expensive and extensive image processing
7 operations. This will be referred to as the vision accelerator
8 or accelerated host processing configuration.

9 Yet another possible configuration involves a complete image
10 processing or machine vision subsystem in the form of an
11 expansion board plugged in the PC's peripheral bus. This board
12 features an on-board CPU, often also coupled with dedicated
13 image processing acceleration hardware. In this configuration,
14 the embedded or target CPU completely off-loads all vision
15 processing operations from the host PC CPU which can now be
16 dedicated to user interface, control, or other tasks running on
17 it concurrently with the image/vision processing tasks running on
18 the vision processor. This multiprocessing configuration will be
19 referred to as the embedded vision processing engine or target
20 processing configuration.

21 A final configuration uses the PC only as the programming
22 and/or user interface environment for a standalone vision
23 processing engine connected to it through a high speed serial or
24 network connection. This configuration will be referred to as
25 the standalone vision processing engine or standalone target

1 processing configuration.

2 Most recently, the PCI bus¹ has enabled high bandwidth data
3 transfers between a plug-in device such as an image acquisition
4 or image processing board and the host PC, further enhancing the
5 performance of all the above mentioned embedded system
6 configurations.

7 The wide availability of PCI compatible components for the
8 PC market makes it also attractive as a local bus to be used
9 inside a processing subsystem such as the above mentioned image
10 or vision processing subsystem configurations.

11 Use of the PCI bus as a local bus is not without its
12 drawbacks however. For example, the PCI bus was designed
13 primarily for the PC market and as such, the topology and data
14 flow is directed from a host CPU through a host bus bridge to a
15 tree of peripheral buses connected through PCI to PCI bridges
16 (herein referred to as peripheral bridges). The host CPU expects
17 to be the only CPU in the system and also expects that its main
18 memory is fixed in the system memory map. Any plug-in card,
19 board or device with an embedded CPU has to ~~allow~~ its local
20 memory to be mapped to avoid conflict with the host PC memory.
21 Memory assignments may come from one or two sources: PCI BIOS or
22 the host operating system (OS). Care must be taken when
23 necessary to avoid system lockups by either hiding devices behind
24 the peripheral bus bridges (where they are inaccessible ~~by~~ the

¹ PCI Local Bus Specification, Revision 2.1, June 1, 1995

1 host PC system), or by temporarily disconnecting them from the
2 PCI bus until they are ready to accept bus transactions.

3 A further drawback to using the PCI bus as a local bus
4 inside PCI based systems is caused by a video graphics adapter
5 (VGA), herein referred to as the display controller, and the
6 discovery process used by the PCI BIOS and host Operating System
7 (OS) to determine the system display controller. There are many
8 different PCI BIOS manufacturers using different discovery
9 algorithms some of which become confused by more than one display
10 controller on the PCI bus. Also, PCI adapter card manufacturers
11 who write drivers for their products sometimes introduce
12 problems, e.g. when an expansion card has a peripheral bridge
13 chip.

14 SUMMARY OF THE INVENTION

15 This invention accordingly provides the following
16 improvements over the current state of the art:

17 1. A flexible vision processing architecture that uses
18 the PCI bus as the embedded processor local bus and which enables
19 a variety of embedded and standalone vision system configurations
20 to be implemented with either no or only minor hardware and
21 software modifications.

22 2. Two different methods for hiding PCI devices that avoid
23 conflicts with the host PC and allow for the embedded CPU to boot
24 correctly.

1 3. Use of a peripheral bus register to reset the embedded
2 CPU when the secondary PCI bus host bus bridge fails to respond,
3 without affecting other secondary bus PCI devices and the host or
4 primary CPU.

5 4. A method of loading the embedded CPU's local memory
6 with diagnostics and Operating System code without the use of ROM
7 or FLASH memory.

8 These and other improvements over the prior art are realized
9 by a reconfigurable vision or other processing system
10 architecture, which is the subject matter of the present
11 invention. The reconfigurable processing system architecture
12 allows for the implementation of a variety of vision or other
13 system configurations to be implemented on a single device,
14 which, in one embodiment, is preferably a PCI bus add-in
15 extension board with an attached digitizer daughter card attached
16 and electrically connected thereto through a PCI Mezzanine
17 connector, and which is plugged into a personal computer PCI
18 expansion slot. The flexible and reconfigurable processing
19 hardware architecture of the present invention ~~uses~~ the PCI bus
20 as the embedded vision processor CPU's local bus, which not only
21 allows for flexibility in vision system configuration but also
22 allows PCI devices to be hidden from the host CPU to allow for
23 proper system startup. The architecture further permits an
24 embedded vision processing CPU to be re-booted when the ~~secondary~~
25 PCI bus host bus bridge fails to respond without affecting host

1 CPU or other secondary PCI bus peripheral devices. Finally, the
2 architecture of the present invention provides a method of
3 loading an embedded vision system CPU's local memory with
4 operating system and diagnostic code without the use of ROM or
5 FLASH memory.

6 BRIEF DESCRIPTION OF THE DRAWINGS

7 Figure 1 is a the block diagram of an embedded processing
8 engine configuration;

9 Figure 2 is a block diagram of a processing accelerator
10 (with digitizer) configuration according to one embodiment of the
11 present invention;

12 Figure 3 is a block diagram of a processing accelerator
13 configured without a digitizer, according to another embodiment
14 of the invention;

15 Figure 4 is a block diagram of a frame grabber configuration
16 according to yet another embodiment of the invention;

17 Figure 5 is a block diagram of a standalone vision
18 processing engine architecture according to one embodiment of the
19 present invention;

20 Figure 6 is a schematic diagram showing a summary of some of
21 the various different architectural configurations from the
22 previous Figures;

1 Figure 7 is a flow chart showing the steps to accomplish the
2 boot time reset; and

3 Figure 8 is a schematic diagram of a system for implementing
4 the bus memory reservation according to another feature of the
5 present invention.

6 DESCRIPTION OF THE PREFERRED EMBODIMENT

7 The hardware described in this invention comprises, in the
8 preferred embodiment, a PCI bus add-in extension board 100 that
9 includes a digitizer subsystem D embodied, for exemplary
10 purposes, in a digitizer daughter card attached and electrically
11 connected to extension board 100 through a PCI Mezzanine
12 connector² (pmc) 36; although such an implementation is not a
13 limitation of the present invention.

14 Although the present invention will be explained
15 utilizing the PCI bus for exemplary purposes, this is not a
16 limitation of the present invention. For example, the compact
17 PCI bus specification which in one implementation is a 64 bit bus
18 operating at 66 MHz and any other PCI bus specification to be
19 specified in the future are considered to be within the scope of
20 the present invention. In addition, any flexible architecture
21 which utilizes a bus structure which makes two (2) devices look
22 like they are connected together including a bus which allows for

² Draft Standard Physical and Environmental Layers for PCI Mezzanine Cards: PMC, IEEE 1386/Draft 2.0, April 4, 1995

1 bus extensions with or without electrical buffering, and which
2 bus and bus extension are designed to operate as a single bus, is
3 considered to be within the scope of the present invention.

4 On the board 100 is a PCI to PCI bridge³ 25, which
5 interfaces a host CPU PCI peripheral bus 27 to a secondary,
6 vision processing system PCI bus 26 included on the PCI extension
7 board.

8 The extension board 100 also includes a processing
9 accelerator subsystem A, implemented preferably as an ASIC
10 (application specific integrated circuit) 20, a VGA (video
11 graphics adapter) or similar display controller 33, and a system
12 CPU subsystem C. The components of the system CPU subsystem C
13 include an embedded system CPU 16, which interfaces to the host
14 system PCI bus 27 via a CPU subsystem CPU bus 15, a host bus
15 bridge 17, and the PCI to PCI bridge 25. Host system components
16 which may be included in the subsystem include local DRAM
17 (dynamic random access memory) system memory 22 and system
18 peripherals 21 such as a DUART RS232 serial controller, NVRAM
19 (non-volatile random access memory) with a real-time clock and a
20 CPLD (complex programmable logic device) providing registers, CPU
21 startup information and other board "glue" logic, as is well
22 known in the art. Power is continually monitored by a CPU
23 supervisory chip, which provides advanced warning to the CPU in
24 the event of a power failure and further provides for manual

³ See PCI to PCI bridge architecture specification Rev. 1.0

1 board reset.

2 In the case wherein the present invention is implemented in
3 a vision system, digitized video data is sent from the digitizer
4 12 (a true data digitizer 12 in the case of an analog camera 14,
5 or a camera interface 12 if the camera 14 is a digital camera) or
6 other general purpose interface over the secondary, vision system
7 PCI bus 26 to local DRAM 22 on the board or to the host PC's
8 local display controller 35 having its own memory 37. The vision
9 processing accelerator 20 "snoops" these transactions using snoop
10 handshake, 21, thus loading the ASIC memory 18a and 18b with the
11 digitized video information. After processing in the processing
12 accelerator 20, the data is sent to the embedded vision system
13 CPU 16 for further processing. Finally, the data may be
14 displayed on local display 28 from the local display controller
15 33 or as an overlay in the host PC's display controller.

16 A plurality of boards can be mapped anywhere into PC memory
17 within the host PC resource constraints and will not cause boot
18 time conflicts with the PC hardware.

19 FLEXIBLE VISION PROCESSING ARCHITECTURE

20 The present invention includes the disclosed architecture
21 which allows a wide variety of system variants to be implemented
22 with minimal hardware and software incremental investment.

23 Figure 1 shows the basic configuration for a vision
24 processing engine system embedded in a PC as a PCI expansion

1 board. By using the PCI bus as the local bus of the vision
2 processing subsystem as well as the local bus of the host PC, the
3 present invention takes advantage of widely available components
4 and easily implements alternative vision system configurations
5 through different combinations of the major functional groupings
6 of the base architecture as will be explained below.

7 The main such functional groups shown in Figure 1 are the
8 digitizer (also referred to as the frame grabber or camera
9 interface) subsystem D, the vision CPU subsystem C, and the
10 vision accelerator subsystem A. All these subsystems are
11 connected to each other through the secondary, vision processing
12 system PCI bus 26 which is itself connected to the PC's (Host
13 CPU) PCI bus 27, host CPU 24 and associated memory 32 through a
14 bus peripheral bus interface, which, in this embodiment is a PCI-
15 to-PCI bus bridge 25³.

16 In the embodiment of such a vision system shown in Fig. 1,
17 image data 13 from multiple cameras 14a-14d can be processed by
18 the vision accelerator 20, the embedded vision system CPU 16, or
19 by a host CPU 24 in cases where the vision system ~~is~~ resident on
20 the host CPU peripheral data bus (e.g. as an adapter or
21 peripheral card). A vision system often includes a local display
22 28 and serial and discrete input/output ports 30 to support real-
23 time machine vision operations and intercommunications with other
24 devices.

³ PCI to PCI Bridge Architecture Specification, Rev. 1.0, April 5, 1994

1 For maximum efficiency, the digitizer 12 in this
2 implementation packs multiple pixels (data samples) into a larger
3 word to match the width of the vision processing system PCI bus
4 26. If the memory data bus 19 width and the vision processing
5 system PCI bus 26 width are not the same, the processing
6 accelerator 20 would reformat peripheral bus words to match the
7 width of the memory data bus 19 or the vision processing system
8 PCI bus 26.

9 The digitizer 12, in connection with direct memory access
10 (DMA) master controller 7, moves pixel data samples in high speed
11 bursts into a portion of peripheral data bus address space. In
12 the present invention, this may be one of the image memories 18a
13 and/or 18b, shared vision system CPU memory 22, host CPU memory
14 32, display controller memory 34, or host display controller
15 memory 37.

16 A more detailed explanation of the operation of the embedded
17 vision system including a processing accelerator A is provided in
18 commonly owned Patent Application Serial Number 08/953,772, which
19 is incorporated herein by reference.

20 To enhance the flexibility of this configuration further,
21 the digitizer subsystem D is implemented as a mezzanine daughter
22 card. Utilizing this implementation, a variety of different
23 camera interface boards can be combined with the same vision
24 processing engine board 100. The connection to such a mezzanine
25 card is through a PCI mezzanine connector (PMC) 36.

1 Figure 2 shows a cost reduced version of the embedded
2 vision engine configuration of Figure 1 wherein the embedded
3 vision system CPU 16, host bus bridge 17, system memory 22,
4 system peripherals 26b and display controller are removed to
5 create the accelerated host configuration mentioned earlier.

6 In the embodiment of Figure 2, the vision processing system
7 comprises a digitizer subsystem D, which includes substantially
8 the same components discussed above with respect to the digitizer
9 subsystem of Figure 1. (The common components of the two
10 digitizer subsystems are identified using the same reference
11 numerals.) The second component of the vision processing system
12 of Figure 2 is the vision accelerator subsystem A, which, like
13 the digitizer subsystem D, is substantially identical to the
14 processing accelerator subsystem A of Figure 1 and common
15 components are indicated using the same reference numerals. This
16 embodiment can be implemented using a plug-in expansion card in
17 much the same manner as the embodiment of Figure 1. However, the
18 expansion card would not include the components of the vision CPU
19 subsystem C (of Figure 1). In this embodiment, the host CPU 24
20 is tasked with running the vision subsystem application.
21 However, the vision accelerator subsystem would still be
22 available with its processing accelerator ASIC 20, which would
23 allow digitized video information to be pre-processed before it
24 is routed to the host CPU 24 for further processing.

25 Figure 3 shows a further cost-reduced variation of the

1 above-mentioned processing accelerator configuration. This
2 configuration incorporates only the processing accelerator
3 subsystem A without a digitizer subsystem. Here, images are
4 acquired through a separate frame grabber board (not shown). As
5 in the configuration of Figure 2, the host PC CPU 24 runs the
6 vision application. However, the vision processing accelerator
7 ASIC 20 is still available to accelerate vision computations.

8 As Figure 4 further shows, a frame grabber (digitizer) D',
9 which can be built from the mezzanine card components by changing
10 the PCI mezzanine connector (36 of Figure 1) to a standard PCI
11 edge connector (not shown), may interface directly to the host
12 CPU PCI peripheral bus 27. This further allows for the
13 development of software drivers from existing software
14 components.

15 Turning now to Figure 5, a fifth embodiment of the invention
16 is shown. In this embodiment, an Ethernet controller 25c or any
17 other communications means such as PPP (TCPIP protocol over a
18 serial line), can be attached to the vision system peripheral bus
19 26 in place of the PCI-to-PCI bus bridge 25 of Figure 1, so that
20 the vision processor acts as a standalone vision processor. As
21 can be seen in Figure 5, the standalone vision processor
22 configuration includes most of the same components described
23 above with respect to the vision system of Figure 1.

24 Other configurations can be envisioned with different
25 functional groupings of the basic subsystems highlighted in

1 Figure 1, including for example, but not limited to,
2 incorporating a camera in the same package as the vision
3 processing engine to form a "smart camera" configuration.

4 A summary of the preceding Figures is shown in Figures 6a-
5 6e, wherein block 'C' represents the Embedded CPU Subsystem
6 including the CPU, host bus bridge, system memory, peripherals,
7 Ethernet controller and display controller; block 'D' represents
8 the Digitizer and Field I/O mezzanine card (100d having a PCI or
9 other system bus compatible edge connector 150 such as a PCI
10 mezzanine connector); and block 'A' represents the processing
11 accelerator and its captive memories. "H" represents a host
12 computer, which is accessed via an Ethernet or PPP connection.

13 The key challenges in making this architecture work are
14 highlighted in the following sections.

15 HIDING PCI DEVICES

16 Up to 21 PCI devices may be addressed on a PCI bus. When
17 devices lie behind a peripheral bridge on its secondary bus, the
18 bridge architecture specification will allow up to 16 devices to
19 be addressed. Secondary bus addressing is implemented by
20 connecting the device IDSEL line to a PCI address line in the
21 range AD[31:16]. When a configuration cycle is issued for a
22 device on the same bus as the target, up to 21 devices may be
23 addressed on AD[31:11], the remaining lines being used to

1 identify configuration cycle type, configuration register number
2 and device function number. This means that a device on the
3 secondary bus (side) of the peripheral bus bridge 25 and
4 connected to AD[15:11] is hidden from the primary side of the
5 peripheral bus bridge 25, but not from the embedded CPU on the
6 secondary PCI bus.

7 The display controller that is used as a local (secondary)
8 display device confuses both the PCI BIOS (Basic Input Output
9 System) and the installed host PC's display controller drivers as
10 to which display device should be used as the system display
11 device. Hiding this local display device behind the PCI bridge
12 (by connecting its IDSEL line to a PCI address line in the range
13 AD[15:11]) means that the display device is only addressable for
14 PCI configuration cycles by the embedded CPU.

15 Although this method provides a way to initialize a private
16 device behind a peripheral bridge, the device is always visible
17 in the final system memory map if the peripheral bridge's memory
18 window encompasses the device's assigned memory region.

19 Another method to hide PCI devices addresses power-on issues
20 (e.g. when the PCI BIOS tries to interrogate the host bus bridge
21 before its initialization sequence is complete it could cause the
22 host PC to "hang" during boot).

23 When a PCI device has multiple clock domains the other
24 clock(s) may be supplied from a source that has a long startup
25 latency for a variety of reasons. The device may also require

1 all clocks to be operational before it can reset its internal
2 state machines. If this delay time is greater than the time that
3 the PC needs before it starts executing its BIOS code, there is
4 the potential for deadlock. Here, a state machine (its operation
5 shown in Figure 7) is used to control the reset release and the
6 PCI device's IDSEL connection sequencing, to ensure correct
7 operation - effectively disconnecting the device from the PCI bus
8 until it is ready to accept bus transactions.

9 In state S0, step 110, the state machine 100 asserts cold
10 CPU reset, asserts host bus bridge reset, disables IDSEL to the
11 host bus bridge, asserts warm CPU reset and resets an 8-bit
12 counter to a count value of 0 (counter reset). In state S1, step
13 120, which is reached when power reaches a sufficient level or
14 manual reset is released, the state machine continues to assert
15 cold CPU reset and host bus bridge reset. It releases the
16 counter reset to allow the counter to increment, and continues to
17 disable IDSEL and assert warm CPU reset. Progression to state S2,
18 step 130 will happen when the counter value reaches 255 (256
19 clock cycles after S1), and the state machine de-asserts cold CPU
20 reset. However, host bus bridge reset remains asserted as does
21 warm CPU reset. In addition, IDSEL remains disabled. Thus, the
22 host CPU can load its mode information and start its system
23 clocks.

24 In state S3, step 140, the state machine continues to de-
25 assert cold CPU reset. It also de-asserts host bus bridge reset,

1 thus allowing the bridge to complete its internal initialization.
2 IDSEL remains disabled and warm CPU reset remains asserted.
3 Moving on to state S4, step 150, IDSEL is enabled, thus allowing
4 host bus bridge to become visible on the secondary system (vision
5 system in this implementation) and visible to the host PC. (Cold
6 CPU reset and host bus bridge reset remain de-asserted and warm
7 CPU reset remains asserted.)

8 Progression from state S4 to state S5 will generally not
9 occur immediately, the average wait time being 128 clock cycles.
10 The 8 bit counter is enabled in all states except state S0, and
11 will continually count with a modulo of 256. There is a minimum
12 wait of one cycle possible which is enough to perform a soft
13 reset on the CPU.

14 Finally, in state S5, step 160, warm CPU reset is de-
15 asserted under software control by deasserting a "halt" register
16 bit with the CPU. Thus, the embedded (vision) system CPU is
17 allowed to boot. This state progression thus allows two CPU's
18 and multiple, potentially conflicting peripheral components to be
19 sequentially initialized without causing system hang-ups.

20 In summary, the Boot Time State Machine 100 performs three
21 functions:

- 22 • 1. It waits for all the clocks that are needed by the
23 device to be active;
- 24 • 2. It asserts device reset until after the clocks are
25 present; and

- 3. It holds IDSEL disabled until a programmable number of PCI clock cycles after reset is deasserted.

The first feature above solves the problem that the embedded CPU must provide a synchronous clock to the host bus bridge which operates at a different frequency from the oscillator that supplies the embedded CPU. Therefore, the CPU frequency generator must supply the host bus bridge clock. This supplied clock is started only after the embedded CPU has loaded its mode information from the complex programmable logic device CPLD. Reset must be deasserted only after the clock from the CPU is present so that the host bus bridge can correctly initialize its internal state. PCI accesses should not be responded to until enough PCI clock cycles have passed from reset deassertion to ensure that the reset completes correctly.

This solution is also equally applicable to a PCI device on the primary or secondary side of a peripheral bridge and anywhere in the PCI bus hierarchy.

Although this method provides a way to initialize a private device behind a peripheral bridge, the device is always visible in the final system memory map if the peripheral bridge memory window encompasses the device's assigned memory region.

EMBEDDED CPU RESET

When the host bus bridge device locks up the secondary PCI bus, the embedded CPU and its devices are inaccessible from the primary PCI bus, including any device registers that can assert a soft reset to the CPU.

A back-door signal (26d of Figure 1) was created from the peripheral bridge to a system peripheral that ensures resets can be asserted to recover the system. While this is useful primarily for debugging, its use in the driver ensures that no problems occur during driver loading. The signal is attached to the peripheral bridge's general purpose input/output (GPIO) port which is addressable from configuration space on the primary bus.

LOADING OS AND TEST CODE

An embedded CPU is usually tested and booted by running code that is present in an on-board ROM or FLASH memory because there is no other way to load this code into the embedded CPU's memory.

With this architecture, code can be loaded over the PCI bus from any source, including an Ethernet device or a host PC. This allows the ROM or FLASH memory to be removed from the board which reduces cost and allows easy field updates by disk or Internet/Intranet.

An additional feature of the present invention facilitates and allows for the use of a non-standard or non-recognized PCI bus connected device such as, for example, a vision system board,

1 embedded CPU or other similar device. When the main system BIOS
2 (the PC (personal computer) boots, the BIOS scans the bus for
3 devices connected to the bus, and allocates system memory for
4 each device. If the BIOS sees a device that it does not
5 recognize, the BIOS ignores the device and does not allocate
6 memory for that particular device.

7 Accordingly, the present invention features a system
8 and method for reserving memory in such a situation; such a
9 system and method being completely independent of the type of
10 BIOS and the type of device plugged into the PCI or other bus.
11 The system and method involves making the otherwise foreign
12 device look like a very standard device so that the BIOS will
13 then reserve memory for the device.

14 For example, in the case of a host or main processing system
15 200, Fig. 8, such as a PC, the host system BIOS is called upon
16 startup of the host or main system 200. The BIOS 202 reserves
17 and allocates system memory resources 206 for each device it sees
18 on the bus 204 (for example, the PCI bus in the preferred
19 embodiment). The BIOS 202 allocates system memory resources 206
20 by addressing each device 210 on the bus 204 and reading the
21 contents of the class code register 208 of each of the devices
22 210 (boards, etc.) on the bus 204. In the preferred embodiment
23 described herein, the memory 206 is located on a peripheral bus
24 bridge 214 (25 in Fig. 1). Each device's class code register 208
25 includes a class code and sub-code that is predefined in the BIOS

1 specification, and which tells the BIOS exactly what type of
2 device is connected to the bus and how much and/or what type of
3 memory to allocate for that device.

4 Accordingly, the system and method of the present invention
5 includes placing a "dummy" device 210 on the bus 204. The dummy
6 device 210a will include a well known and standard class
7 code/sub-class code in its class code register 208a. The non-
8 standard device 210b, such as a device with an embedded
9 processor, etc., will not have a class code register 208. An
10 example of a well known device is an Ethernet card which every
11 BIOS 202 will recognize.

12 The BIOS 202 will then allocate a specified amount of memory
13 206a to the particular device of interest 210b (such as a non-
14 standard device) based on the class code of the dummy board 210a.
15 Subsequently, the driver 212 for the particular device 210b of
16 interest will determine the address range of the memory space
17 206a allocated to the dummy device 210a, and will divide and
18 allocate that memory range to various elements (embedded
19 processors, registers, FIFO's, etc.) on the device 210b which
20 will need to utilize the allocated memory space 206a.

21 In this manner, the above described feature of the present
22 invention provides for memory reservation by utilizing a dummy or
23 surrogate device to reserve memory for a non-standard device,
24 thus forcing the system BIOS to allocate memory to a desired
25 device based on the class code of a dummy or surrogate device.

1 Modifications and substitutions by one of ordinary skill in
2 the art are considered to be within the scope of the present
3 invention which is not to be limited except by the claims which
4 follow.

5 What is claimed is: